How to Build Up Contemporary Software Professionals
Project-Based Learning in Data Structure and Programming

Magdalina Todorova[1], Hristo Hristov[2], Eliza Stefanova[1], and Nikolina Nikolova[1]

Faculty of Mathematics and Informatics, Sofia University, Bulgaria
[1]{magda, eliza, nnikolova}@fmi.uni-sofia.bg
[2] ico.dimov@gmail.com

Abstract. This article describes project-based learning (PBL) approach applied in the course Data Structures and Programming taught in Faculty of Mathematics and Informatics at Sofia University. PBL is a component of a larger and integral approach that includes teaching through lectures and seminars. The proposed PBL is based on group projects with medium development timelines. The basic phases of projects development are presented. Sample project descriptions in terms of subject are given. The results of the pilot are analyzed and future development requirements are proposed.

Keywords: Project-Based learning, University programming course, Software Professionals Teaching

1 Introduction

"Everything I know, I know how to use and where to apply" – this is the main statement of the modern project-based learning [1]. This idea is attractive for a number of educational system designers who strive to achieve a balance between academic knowledge and practical skills. PBL is not a new concept in pedagogy. It was developed during the second decade of the last century by J. Dewey and C. Kilpatrick. The great popularity of the method is due to good welding between theoretical knowledge and its practical application ([2], [3]). PBL was introduced experimentally for the 2009/2010 academic year in Faculty of Mathematics and Informatics (FMI), Sofia University. It was applied to one of the introductory programming courses – namely "Data Structures and Programming" (DSP). It was a part of a complete educational approach, consisting of traditional lecture-based education, additional seminars and project-based learning.

Why did we decide to implement PBL approach?

Prior to the presented experiment, we used to apply the traditional style of teaching - we would deliver the theory as lectures and after that students would practice particular tasks during the seminar through drills, reproducing solutions in the computer labs. There were some positive results: the students were good enough in the chosen integrated development environment (IDE) and comparatively complex algorithmic problems. These positive sides, however, were not enough for their practical realization as software developers.

Furthermore, there is a practice some students to work as part-time developers in a software company. Our observation is that these students achieve better results in the subject DSP. Besides the academic knowledge, they have an opportunity to master practicing, to develop their skills in working on a project or team working. The relation between theory and practice motivates these students to be more disciplined and hardworking and to take more from the lectures and seminars.

These concerns, combined with our strife to produce highly-qualified software developers at both analytical and application level, were our motivation to give such an opportunity to other students as well. We choose to implement the PBL approach, because it gives a way to organize a learning environment as close to the real work in a software company as possible. We designed the PBL-based course as part of the lab sessions as a preparatory step for their future work as a software developer.

The application context of the PBL approach

"Data Structures and Programming" course in FMI is taught to Informatics students. The programme includes a compulsory DSP course with 45 academic hours of lectures and 30 academic hours of seminars, distributed in 15-week term. The lab sessions are a separate elective course with 30 academic hours. Although the latter is elective, about 80% to 90% of the students attended it. During the pilot of PBL, 62 students attended the course. It took place in a computer lab, with a personal workplace for each student.

The DSP course deepens and broadens the students' knowledge and skills acquired in the preceding courses: Introduction to Programming and Object-Oriented Programming (both based on C++ programming language). That is why it was natural the DSP course to be based on C++. The topics covered by the compulsory DSP course and corresponding labs elective course are: data structures – definition, classification; simple data structures; linear data structures (stack, queue, deck, linked list; circular linked list); binary tree, binary search tree, balanced binary tree, perfectly balanced binary tree, B-trees; graphs, paths inside graph; heterogeneous data structures; files; algorithms for searching and sorting; hashes, applications.

2 How was project-based learning applied to the course?

Some elements of the PBL on Data Structures and Programming in FMI were borrowed from the project-based learning approach to programming in the Pennsylvania State University [4]. In our case, all students were divided into 30 teams, and had to choose among 10 topics for a project assignment, formulated by the lecturers in such a way that they were approximately equivalent in complexity and volume.

The training proceeded in three phases.

First phase: *Project preparation*.

At that phase lecturers and tutors are more active. Main steps in the phase are:

a) Choosing a project topic, formulating the goals and tasks for every project

Defining suitable topics for project assignments was a matter of extreme importance, in which we were guided by the following requirements: the topics should have covered as much as possible of the lecture course contents; the students should have been given the opportunity to apply knowledge and techniques learned during the preceding courses in programming, discrete mathematics and information technologies; the topics should have been related to real-life practical problems; the goals and tasks should have been formulated in a clear and accurate manner; the topics should have been consistent with the knowledge needed for the project fulfillment.

b) Formulating the project assignments

In accordance with the formulated goals and tasks of every project, the training material underwent a thorough selection procedure, resulting in a list of basic concepts and problems. Next, the application field of each assignment was chosen, in turn conditions and requirements of the project specified accordingly. Whenever an assignment contained new concepts and algorithms, respective literature references were recommended and additional training materials were provided.

c) Choosing the team members

The number of partcipants per project team depended on the scope and complexity of the project assignment. The projects, defined for the sake of the experiment, involved two or three students each. Every team was assigned a team leader. Once the teams were formed and the projects were assigned, the teams were introduced to the project goals, tasks and expected results; the terms for project implementation; the requirements for drawing up the source code documentation; criteria for project assessment; projects presentation; communication among the team members, and between the team leader and the lecturer, etc.

Second phase: *Project implementation.*

Although the organization of the PBL started in the very beginning of the 15-week term, the duration of the actual work on the projects was about 6 to 7 weeks. The first weeks students gained knowledge necessary for realization of their projects. At first, the teams spent time in clarifying the projects goals and tasks, and got familiar with the additional learning materials, algorithms, technologies, etc. For this purpose, consultations using the provided literature references were carried out by the lecturer via email communication. Later on, the teams focused on the comprehension of the algorithms for solving the problems in the project assignments, and specified the technologies that were to be used both for the projects implementation and for the final presentation. The team leaders distributed the tasks among the team members and elaborated on schedules for meetings and discussions. The internal team communication was accomplished via the Moodle learning management system (LMS) [5] and email. The most important decisions taken during the team discussions were included in the final project presentation.

Third phase: *Final activities.*

The final activities on the project work were:

a) Project presentations

This was an official meeting when the teams present, analyze and assess the results of their work on the project. We conducted this meeting in the following manner: The members of each team were given 15-20 minutes to explain the solution of the problem in their assignment, the structure of the software program, the algorithms and classes in use (both standard and personally elaborated for the sake of the project), as well as the rationale behind making the particular choices. The students demonstrated the program using input data prepared in advance, as well as using data offered by the lab assistant. During the presentation, some of the teams were asked to make small additions or changes in the project functionalities, which they were supposed to accomplish in situ for less than an hour.

b) Projects analysis and assessment by the lecturer

Students were informed about the criteria for project assessment in advance (in %):

- Correct solution of the problem and effectiveness of the algorithms – 50%
- Well defined structure of the classes used for solving the problem – 20%
- Good coding style – 10%
- Adequate and clear source code documentation – 10%
- Project presentation – 10%

The criteria were applied only if the examined students were able to present their projects convincingly and/or to quickly implement the small changes required by the tutor. The latter condition was considered indicative of whether the project was genuine. In case of inability to meet this condition, the project was not approved. Moreover, one of the requirements for elaborating the project documentation included summary (report) of the teamwork during the project implementation. These reports had to present each student's personal share of the accomplished work and respective self-evaluation, and they were later taken into consideration when the lecturer assessed the projects.

On the basis of the presentation and the analysis of the work done, and using the formulated assessment criteria, the lab assistants elaborated on the scores for all projects assignments.

3 Examples of project descriptions

The projects used in the course were based on the following criteria:

- comprehensive – as much as possible lecture material to be covered;
- real world problem described (a simplified version of a specific task);
- knowledge in other learning disciplines required;
- team work development is implied.

All course projects had common requirements:

- to master programming skills for working with C++ input/output system;
- to learn basic methods for data representation in the computer;
- overtaking „from abstract to physical data representation" barrier;
- to choose procedure-oriented or object-oriented approach to solve the problem;

- to follow principles of separate compilation in the development environment.

In this paper, we shortly present an excerpt of three projects with these common requirements applied. Their full description, as well as the description of the other projects used in the course could be found in the course LMS [5].

The projects listed below differ in the scope and deepth of knowledge required to solve the problem.

3.1 Project "Interpreter"

This project is the most abstract one. It requires:

- Knowledge from other disciplines (Discrete Mathematics, Discrete Structures):
  - understanding and using both grammar and language derived from grammar;
  - clearing the left recursion from grammar production rules;
  - normal forms of context-free grammar;
  - building a parsing tree of words produced from context-free grammar;
- argumentation and implementation of appropriate data structures:
  - array/linked lists;
  - typed binary tree.

Project description presented to students is given on the figure below (fig.1).

The programs in the teaching programming language EXPR are sequences of operators for assignment (=) and output. Every row of an EXPR program can have one operator at most – either for an assignment or output. The variables in the EXPR programming language consist of low Latin letters (a-z). Beside this, the language offers positive natural numbers in unsigned long int domain. An assignment operator allows variables to have either numeral value or value of a simple arithmetic expression. Respectively, the output operator allows output of a variable value or number. Every single row in our EXPR program can be described by the following context-free grammar G = <N, T, Line, P>. Its production rules are as follows:

```
Line  -> Var = Num | Var = Expr | print Var | print Num
Var   -> a | b | ... | z | aVar | bVar | … zVar
Num   -> 0 | ... | 9 | 1Num | ... | 9Num
Expr  -> Expr + Term | Expr - Term | Term
Term  -> Term * Factor | Term / Factor | Factor
Factor -> Var | Num | (Expr)
```

***Your task** is to write an interpreter of the EXPR language. There are certain input/output requirements for your program [5].*

Fig. 1. Short description of a project Interpreter.

3.2 Project "Vehicle database"

The second project is the most practically oriented. It requires:

- knowledge from other subjects (Introduction to Programming, Object-Oriented Programming):
  - algorithms for searching and sorting;
- specific implementation of text based/binary based file input/output;
- argumentation and implementations of appropriate data structures:
  - linked list (single or double linked, sorted);
  - binary search tree;
- understanding the concept of secondary data storage (disk based swapping).

Project was presented to students through a description as shown on fig.2.

Your task *is to write a C++ program, which serves as simple vehicle database system. Vehicle data is stored in a disk file (text or binary – record based). Every record consists of the following data: registration number; current mileage in kilometers; average fuel consumption per 100 km; vehicle price; vehicle brand; vehicle type – automobile, truck or bus. For trucks there is also payload in tons and for buses there is a limit of passenger places available (both seats and standing). Your program should have the following functionality:*

(a) to create a binary file with the records specified. The full description of data format is given in [5] and it depends on the type of the vehicle.

(b) to support a list of registration numbers, type and brand of vehicle, ordered by fuel consumption for given mileage; Program should be able to display this list;

...

(g) to calculate the average passenger places for buses (both seating and standing).

Fig. 2. Short description of project Vehicle database.

3.3 Project "Huffman"

The third project is the most difficult. Its goals are similar to those of the project described in 3.2. Searching and sorting algorithms are replaced by the bitwise operators. This project requires different data structures in order to be solved. These data structures include: tables (static or dynamic) and encoding/decoding binary trees.

Project description presented to students is given on figure below (fig.3).

Your task *is to write a program which reads single or many strings from a given text file. For every string read (or all strings as a whole), to build Huffman frequency table and Huffman tree based on that table. Using the tree built in the previous step, to encode input string(s) and to output its binary code. Your program should also calculate compression ratio (the original size in bytes divided by the encoded size in bytes). Using an existing binary code and Huffman tree (already built) your program should be able to decode original input string (or strings).*

Fig. 3. Short description of a project Huffman.

4 Project implementation process. Feedback

The common components for all projects (classes, algorithms, approaches and techniques) were explained and developed during the lab session under the lecturer's supervision. Using the object-oriented approach, the students implemented the main dynamic data structures (list, stack, queue, priority queue, tree and graph), and examined, implemented and tested different algorithms for depth-first and breadth-first tree and graph traversal, which were not typically included in the lecture course. The students mastered a series of programming techniques, namely working with pointers, memory management, object serialization, exception processing, and others.

Furthermore, every team adapted parts of the components developed for the purpose of their team projects. Part of the particular work on a given project was done during the lab session under the guidance of the tutor, while the rest was accomplished independently.

In the inquiry which followed the presentation and evaluation of the projects, the students shared their satisfaction with the adopted form of teaching. Some of them described the results from the project implementation as 'the first reasonable and practically applicable training' they have got; some shared that they had finally seen real-life applications of the Discrete Mathematics; while others expressed their satisfaction of having overcome multiple challenges in processing of large data sets without suffering the overflow effects or using recursion. It was the opinion of most of the students that their involvement in the project had helped them to more easily comprehend the lecture course contents, to improve their work efficiency, to develop their programming style, and taught them to develop software documentation, along with shaping their idea of the carrier they had chosen. Also, it was a common opinion that commenting the source code and algorithms in the case of the teamwork had proven to be an extremely useful experience.

Thus, based on the project presentations, on the observation, conversations and consultations during the projects development, and on the information obtained from the questionnaires, we confidently estimate the conducted training experiment as a success. In support of this statement there are several arguments:

- The students who got involved in the project-based training not only achieved higher scores as a result (fig. 4), but they also attained more profound and solid knowledge.
- They became more active participants in the process of tuition.
- The students exhibited desire and drive to discovery.
- Closer cooperation was built between students and lecturers, which allowed the lecturers to better understand the students' hardships in comprehending the material.

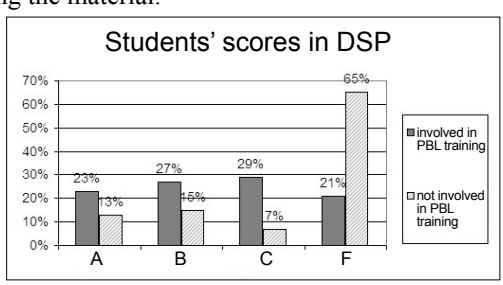

Fig. 4. Comparison of the scores of students who ARE and who are NOT involved in the PBL experiment.

During the application of PBL approach a series of challenges arose.

The main difficulty was related to the students' distribution into working teams. In certain cases, some members of the teams had not taken equal shares of the project development, yet they were equally assessed according to the predefined evaluation criteria. The lecturers' team sought solutions for overcoming these obstacles in due course, and the experience gained is to be taken into consideration when applying this approach during the next academic year.

In addition, the tutor team gained the experience that not all the projects are appropriate for that level of development of students. First and second projects presented in the paper were actually at the same level of difficulty (as we had presumed), but the third project was impossible to complete by students in time-frame of the course. This conclusion will be taken into account in the next year preparation of the projects.

5 Conclusion and future steps

The detailed analysis of the results is just beginning. Despite the problems aroused during the pilot of the PBL approach applied to one of the introductory programming courses, namely Data Structures and Programming the results prove that it was successful. The lecturer team intends to expand and improve the application of this approach. We foresaw the following directions of its further improvement: applying the method to the lecturing and/or seminar form of training; applying it to the other introductory programming courses, namely Introduction to Programming and Object-Oriented Programming; and applying it to the tuition on a specific lecture topic. In order to improve the application of this training approach, we consider to launch collective discussions of the results achieved in the projects, as well as external evaluation (by another lecturer or a software specialist), followed by publishing the best projects on the Internet.

Acknowledgements. The paper is supported by Grant 162/2010 from Sofia University Research Fund.

References

1. Polat E. Project-based learning – article on the site of Russian Academy of Education, http://www.iteach.ru/met/metodika/a_2wn3.php (in Russian) (last visited on 05/20/2010)
2. Sendova E., Stefanova E., Boytchev P., Nikolova N., Kovatcheva E. (2008) IT education – challenging the limitations instead of limiting the challenges, Proceeding of 6th International Conference for Informatics and Information Technology (CIIT 2008), Bitola, Macedonia, 10-14 February 2008
3. Stefanova E., Sendova E., v. Diepen N., Forcheri P., Dodero G., Miranowicz M., Brut M., et al Innovative Teacher - Methodological Handbook on ICT-enhanced skills, Faleza-Office 2000, Sofia, 2007
4. Azalov, P., Richards, D., Project-Based Teaching of Intermediate Programming, International Symposium IGIP/IEEE/ASEE 2004, September 27-30, Fribourg, Switzerland
5. Data Structures and Programming course in Moodle CMS, https://moodle.openfmi.net/ (last visited on 07/07/2010)